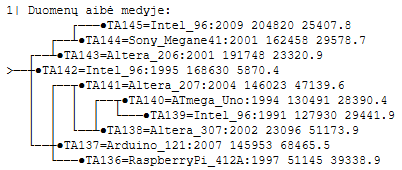
KAUNO TECHNOLOGIJOS UNIVERSITETAS

Duomenų struktūros (P175B014) Laboratorinio darbo Nr.2 ataskaita

Atliko Arnas Švenčionis gr. IFF-8/11

Privalomi metodai:

Pradiniai duomenys:



**Boolean addAll(BstSet<? Extends E>c)**

public boolean addAll(BstSet<?extends E> c){

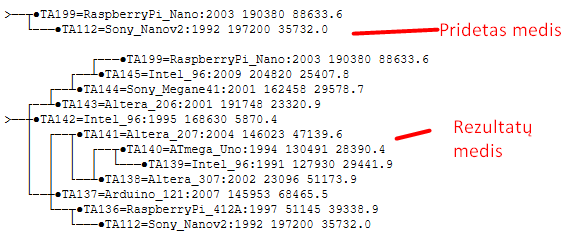
if(c.size() == 0 || this == null) return false;

for (E e : c) this.add(e);

return true;

}

Rezultatai:



**E higher(E e)**

public E higher(E e){

if(e == null) return null;

BstSet<E> temp = new BstSet<>();

temp = higherCloneRec(e, root, temp);

return temp.getMin(temp.root).element;

}

public BstSet<E> higherCloneRec(E e, BstNode<E> node, BstSet<E> n) {

if (node == null) {

return null;

}

int cmp = c.compare(node.element, e);

if(cmp > 0) n.add(node.element);

higherCloneRec(e, node.left, n);

higherCloneRec(e, node.right, n);

return n;

}

Rezultatai:

Pasirinkus elementą TA137=Arduino\_121:2007 145953 68465.5 rezultatas:



**E pollLast()**

public E pollLast(){

if(size == 0) return null;

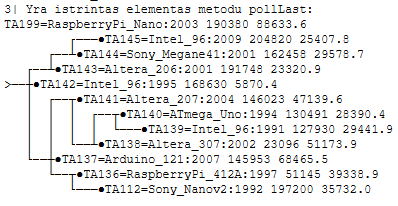
BstNode<E> temp = this.getMax(root);

this.remove(temp.element);

return temp.element;

}

Rezultatai:



**Set<E> headSet(E element)**

public Set<E> headSet(E element) {

if (element == null) {

throw new NullPointerException();

}

Set<E> hs = new BstSet(); //TailSet sub-tree

if (contains(element)) {

BstNode<E> n = root;

headRecursive(hs, n, element);

}

return hs;

}

private BstNode<E> headRecursive(Set<E> Set, BstNode<E> n, E d) {

if (n == null) {

return null;

}

if (c.compare(n.element, d) != 0) {

Set.add(n.element);

headRecursive(Set, n.left, d);

headRecursive(Set, n.right, d);

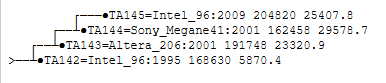
}

return n;

}

Rezultatai:

Pasirinkus elementą TA137=Arduino\_121:2007 145953 68465.5 rezultatas:



**Set<E> tailSet(E element)**

public Set<E> tailSet(E element) {

if (element == null) {

throw new NullPointerException();

}

Set<E> ts = new BstSet();

if (contains(element)) {

BstNode<E> n = getNode(element);

tailRecursive(ts, n);

}

return ts;

}

private BstNode<E> tailRecursive(Set<E> hs, BstNode<E> node) {

if (node == null) {

return null;

}

hs.add(node.element);

tailRecursive(hs, node.left);

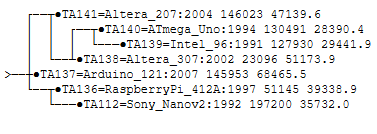
tailRecursive(hs, node.right);

return node;

}

Rezultatai:

Pasirinkus elementą TA137=Arduino\_121:2007 145953 68465.5 rezultatas:



**Set<E> subSet(E element1, E element2)**

public Set<E> subSet(E element1, E element2) {

if (element1 == null || element2 == null) {

throw new NullPointerException();

}

Set<E> temp = new BstSet();

if (contains(element1) && contains(element2)) {

BstNode<E> n = getNode(element1);

headRecursive(temp, n, element2);

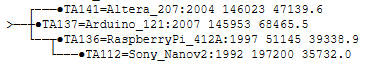
}

return temp;

}

Rezultatai:

Pasirinkus pradinį elementą TA137=Arduino\_121:2007 145953 68465.5 ir galutinį elementą TA138=Altera\_307:2002 23096 51173.9, rezultatas:



**void remove() (Iteratoriaus)**

public void remove() {

stack.remove(parent);

//throw new UnsupportedOperationException("Studentams reikia realizuoti remove()");

}

Rezultatai:

50| Automobilių aibė su iteratoriumi:

51|

52| TA100=C1ATmega\_315M:1997 50000 1700.0

53| TA104=C5Arguino\_Nano:1946 365100 9500.0

54| TA105=C6Arguino\_Uno:2001 36400 80.3

55| TA106=C7RaspberryPi\_Three:2001 115900 7500.0

56| TA107=C8RaspberryPi\_Four:1946 365100 950.0

57| TA108=C9RaspberryPi\_Two:2007 10000 850.3

58|

59| Iteratoriaus remove

60|

61| TA100=C1ATmega\_315M:1997 50000 1700.0

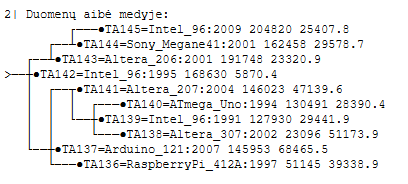
62| TA104=C5Arguino\_Nano:1946 365100 9500.0

63| TA105=C6Arguino\_Uno:2001 36400 80.3

64| TA108=C9RaspberryPi\_Two:2007 10000 850.3

Individualūs metodai:

Pradiniai duomenys:



**E floor(E e)**

public E floor(E e){

if(this.contains(e) && e != null)

return e;

return null;

}

Rezultatai:

Pasirinkus TA137 objektą, atsakymas yra:



**E lower(E e)**

public E lower(E e){

if (e == null) {

return null;

}

BstSet<E> temp = new BstSet<>();

temp = lowerCloneRec(e, root, temp);

return temp.getMax(temp.root).element;

}

public BstSet<E> lowerCloneRec(E e, BstNode<E> node, BstSet<E> n) {

if (node == null) {

return null;

}

int cmp = c.compare(node.element, e);

if (cmp < 0) {

n.add(node.element);

}

lowerCloneRec(e, node.left, n);

lowerCloneRec(e, node.right, n);

return n;

}

Rezultatai:

Pasirinkus TA137 objektą, atsakymas yra:



**SortedSet<E> tailSet(E fromElement)**

public SortedSet<E> tailSetI(E fromElement){

if (fromElement == null) {

throw new NullPointerException();

}

BstSet<E> temp = new BstSet<>();

if (contains(fromElement)) {

temp = higherCloneRec(fromElement, root, temp);

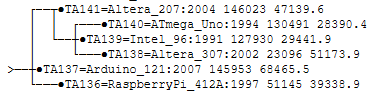
}

return temp;

}

Rezultatai:

Pasirinkus TA137 objektą, atsakymas yra:



**treeHeightR(BstNode<E> node)**

private int treeHeightR(BstNode<E> node) {

BstNode<E> n = node;

if (root == null) return 0;

int hLeft = 0;

int hRight = 0;

if (n.left != null) hLeft = treeHeightR(n.left);

if (n.right != null) hRight = treeHeightR(n.right);

int max = (hLeft > hRight) ? hLeft : hRight;

return max + 1;

}

Rezultatai:



Greitaveika:

Reikia palyginti TreeSet<Integer> ir HashSet<Integer add(Object o) ir remove(Object o) metodus.

Greitaveikos rezultatai:

1| Greitaveikos tyrimas:

kiekis(\*k) addTreeSet addHashSet removeTree removeHash

10000( 1) 0.0171 0.0047 0.0107 0.0023

20000( 2) 0.0163 0.0039 0.0083 0.0011

40000( 4) 0.0164 0.0019 0.0085 0.0011

80000( 8) 0.0281 0.0045 0.0243 0.0035

Bendras tyrimo laikas 2.227 sekundžių

Išmatuotas tyrimo laikas 0.153 sekundžių

t.y. 93.1% sudaro pagalbiniai darbai

Normalizuota (santykinė) laikų lentelė

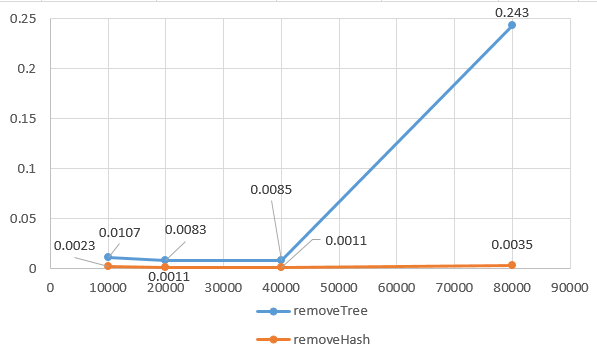
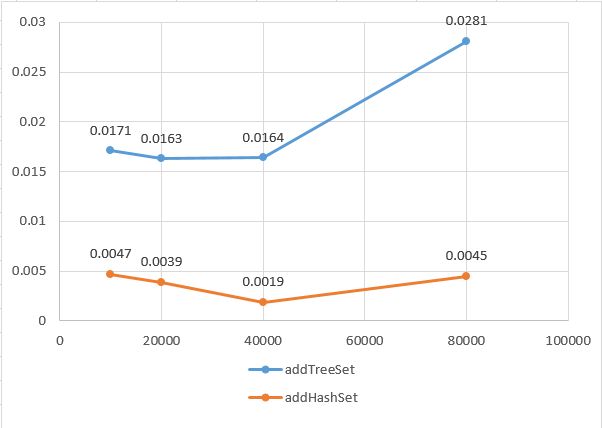
kiekis(\*k) addTreeSet addHashSet removeTree removeHash

10000( 1) 1.00 0.27 0.63 0.13

20000( 2) 0.95 0.23 0.48 0.07

40000( 4) 0.95 0.11 0.49 0.06

80000( 8) 1.64 0.26 1.42 0.20



Išvados:

Laboratorinio darbo metu susidūriau su daug kėblumų. Darbui atlikti reikėjo prisiminti rekursiją, kaip ji veikia, nes ją teko naudoti nevieno metodo realizacijai. Individualūs metodai daug problemų nesukėlė. Sudėtingesni buvo privalomi metodai tokie kaip headSet, tailSet ir taip toliau. Kadangi šį kartą greitaveika veikė per kelias klases, ją išsiaiškinti buvo sudėtingiau. Ko gero įdomiausia laboratorinio darbo dalis buvo graphical interface nagrinėjimas ir koregavimas. Buvo įdomu matyti vieną nemažą sistemą sistemingai dirbant.

Daugiausia laiko praleidau prisimindamas rekursiją ir aiškindamasis kaip veikia patys medžiai. Po to darbas vyko gana sklandžiai.

Apytikslė darbo atlikimo trukmė: Individualūs metodai – 5 val., Privalomi metodai – 10 val., Greitaveika – 1 val., GUI pakeitimai – 4 val.